Distributed Machine Learning

Maria-Florina Balcan, Georgia Tech

Model for reasoning about key issues in supervised learning

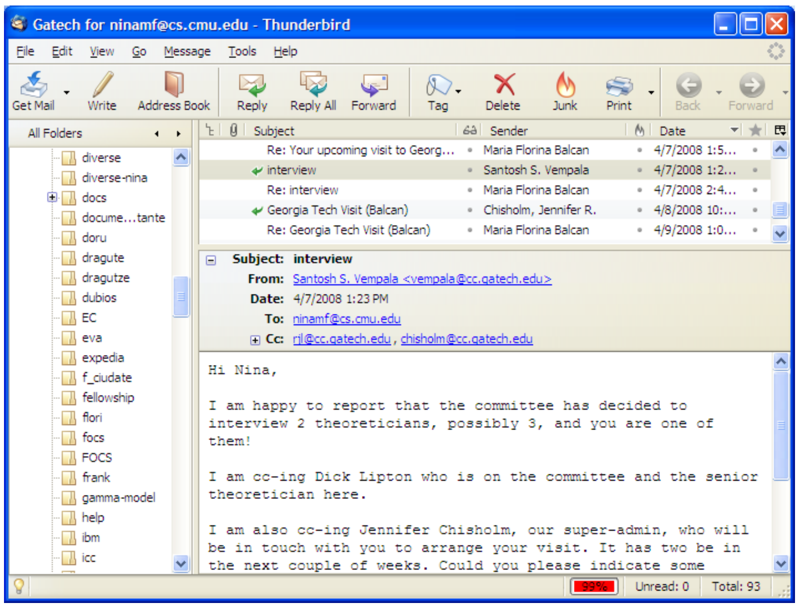
[Balcan-Blum-Fine-Mansour, COLT 12]

Supervised Learning

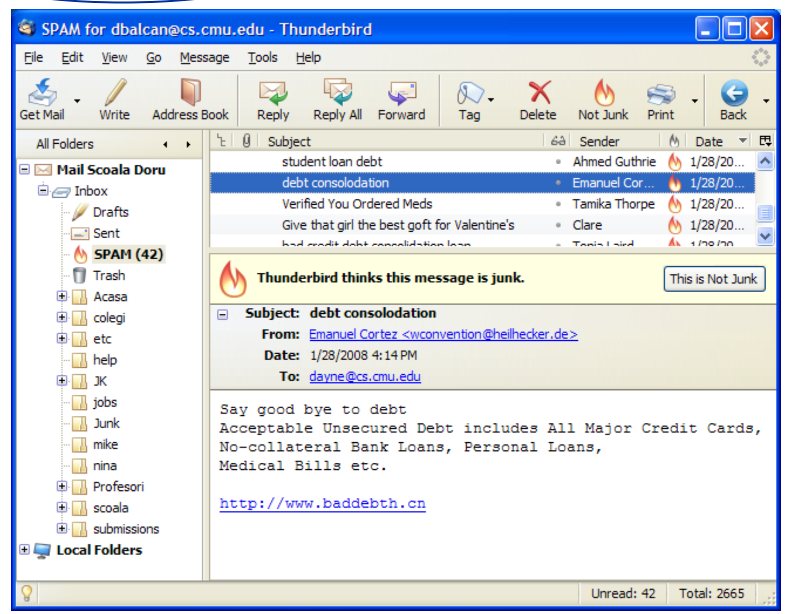
- Example: which emails are spam and which are important.

Supervised classification

Not spam

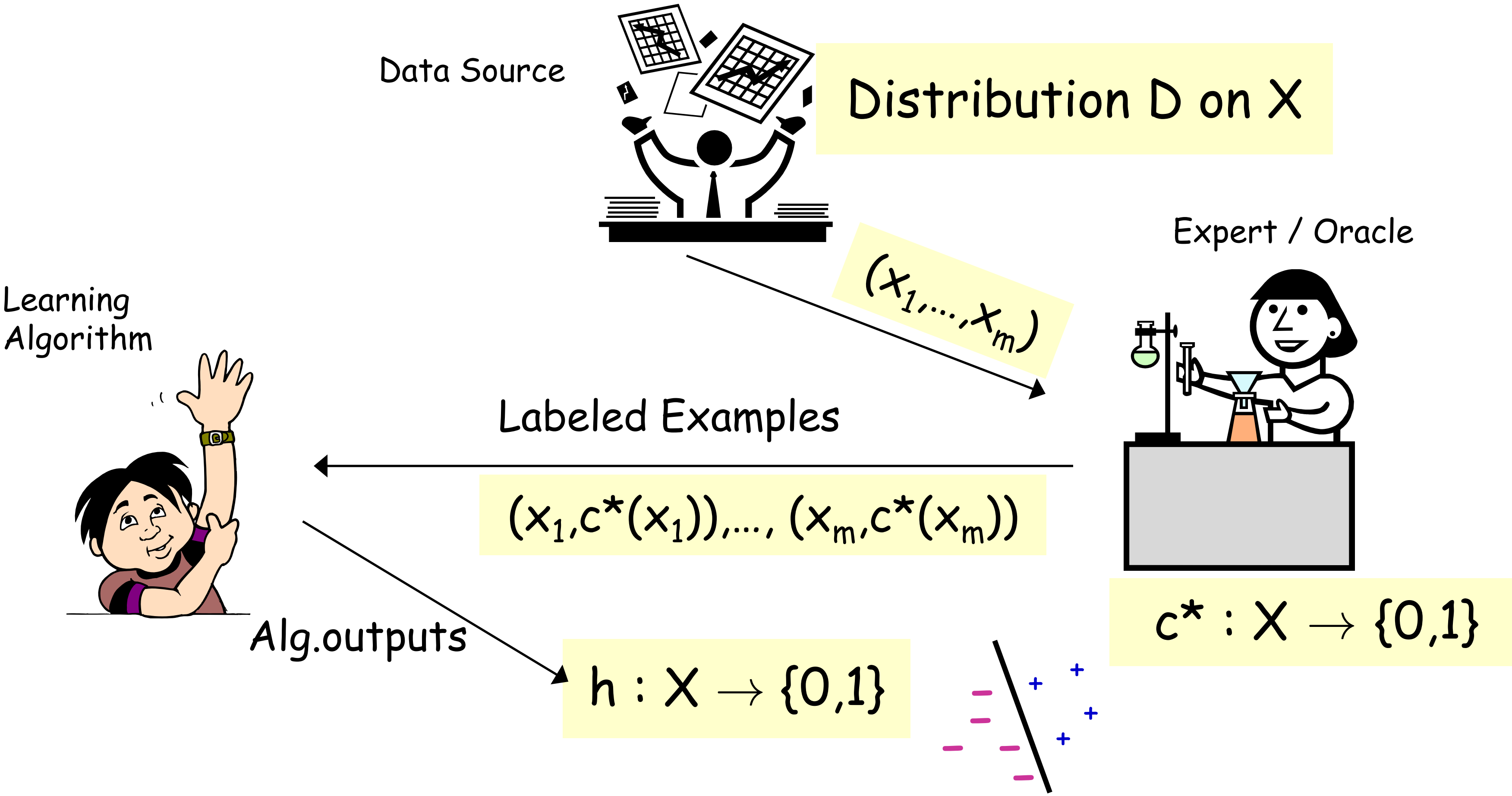


spam

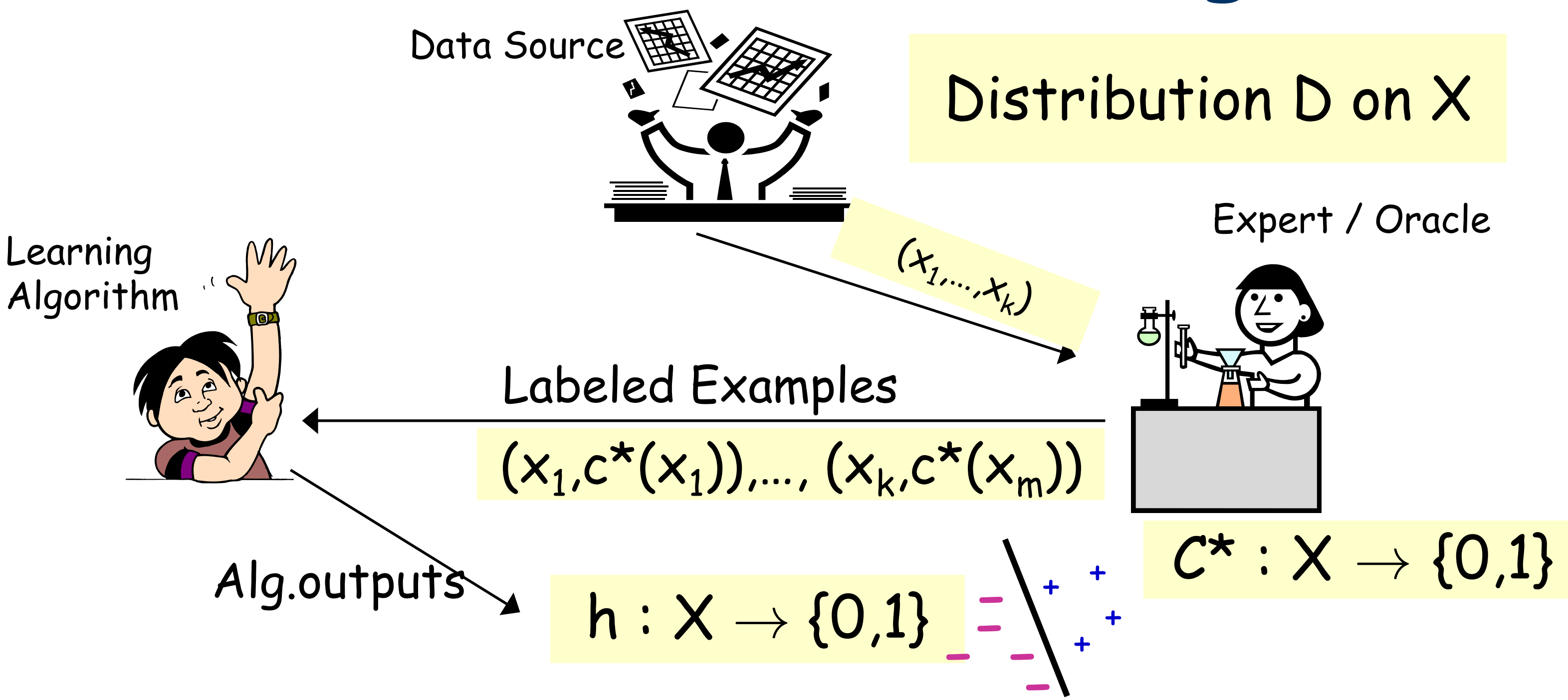


Goal: use emails seen so far to produce good prediction rule for future data.

Statistical / PAC learning model



Statistical / PAC learning model



- Algo sees $(x_1,c^*(x_1)),\ldots, (x_k,c^*(x_m))$, x_i i.i.d. from D
- Do optimization over S, find hypothesis $h \in C$.
- Goal: h has small error over D.

$$err(h)=Pr_{x \in D}(h(x) \neq c^*(x))$$

- c^* in C, realizable case; else agnostic

Two Main Aspects in Classic Machine Learning

Algorithm Design. How to optimize?

Automatically generate rules that do well on observed data.

E.g., Boosting, SVM, etc.

Confidence Bounds, Generalization Guarantees

Confidence for rule effectiveness on future data.

Sample Complexity Results

Confidence Bounds, Generalization Guarantees

Confidence for rule effectiveness on future data.

Theorem

$$m \geq \frac{1}{\varepsilon}\left[VCdim(C)\log(\frac{1}{\varepsilon}) + \ln\left(\frac{1}{\delta}\right)\right]$$

labeled examples are sufficient s.t. with prob. at least $1-\delta$, all $h \in C$ with $\hat{err}(h) = 0$ have $err(h) \leq \varepsilon$.

- Agnostic – replace ε with ε^2.

Distributed Learning

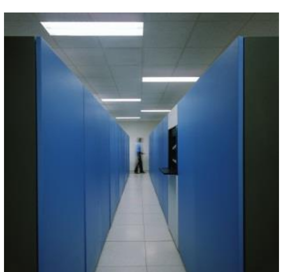
Many ML problems today involve massive amounts of data distributed across multiple locations.

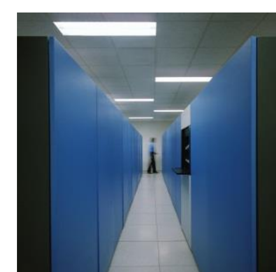
Often would like low error hypothesis wrt the overall distrib.

Distributed Learning

Data distributed across multiple locations.

E.g., medical data

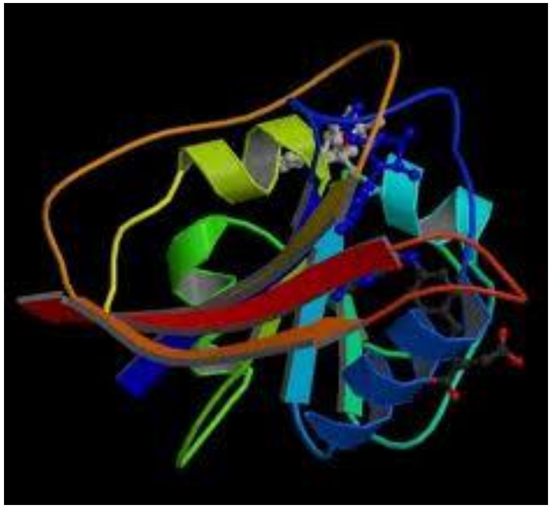




Distributed Learning

Data distributed across multiple locations.

E.g., scientific data



Distributed Learning

- Data distributed across multiple locations.
- Each has a piece of the overall data pie.
- To learn over the combined D, must communicate.

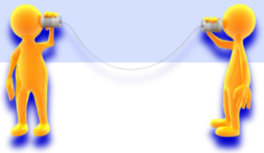
Important question: how much communication?

Plus, privacy & incentives.

Distributed PAC learning [Balcan-Blum-Fine-Mansour,COLT 2012]

- X – instance space. k players.
- Player i can sample from D_i, samples labeled by c^*.
- Goal: find h that approximates c^* w.r.t. $D=1/k\,(D_1 + \ldots + D_k)$
- Fix C of VCdim d. Assume $k \ll d$. [realizable: $f \in C$, agnostic: $f \notin C$]

Goal: learn good h over D, as little communication as possible



- Total communication (bits, examples, hypotheses)
- Rounds of communication.

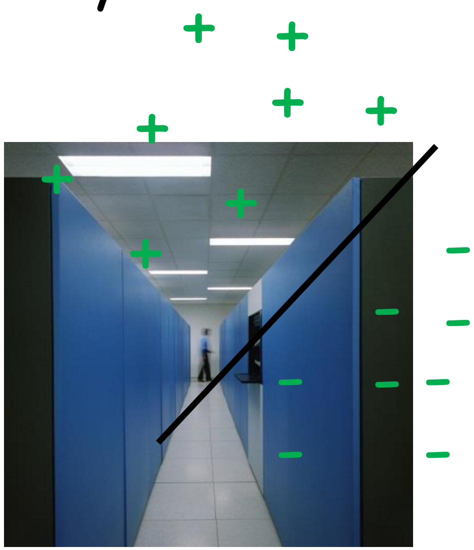
Efficient algos for problems when centralized algos exist.

Interesting special case to think about

k=2. One has the positives and one has the negatives.

- How much communication, e.g., for linear separators?

Player 1



Player 2

Overview of Our Results

Introduce and analyze Distributed PAC learning.

- Generic bounds on communication.
- Broadly applicable communication efficient distributed boosting.
- Tight results for interesting cases (conjunctions, parity fns, decision lists, linear separators over "nice" distrib).

Analysis of privacy guarantees achievable.

Some simple communication baselines.

Baseline #1
$d/\epsilon \log(1/\epsilon)$ examples, 1 round of communication

- Each player sends $d/(\epsilon k) \log(1/\epsilon)$ examples to player 1.
- Player 1 finds consistent $h \in C$, whp error $\leq \epsilon$ wrt D

…

Some simple communication baselines.

Baseline #2 (based on Mistake Bound algos):

M rounds, M examples & hyp, M is mistake-bound of C.

- In each round player 1 broadcasts its current hypothesis.
- If any player has a counterexample, it sends it to player 1. If not, done. Otherwise, repeat.

…

Some simple communication baselines.

Baseline #2 (based on Mistake Bound algos):

M rounds, M examples, M is mistake-bound of C.

- All players maintain same state of an algo A with MB M.
- If any player has an example on which A is incorrect, it announces it to the group.

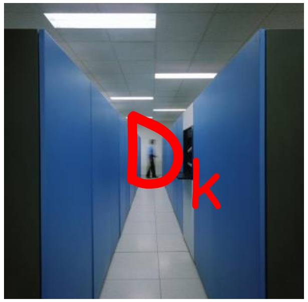
…

Improving the Dependence on $1/\epsilon$

Baselines provide linear dependence in d and $1/\epsilon$, or M and no dependence on $1/\epsilon$.

Can get better $O(d \log 1/\epsilon)$ examples of communication!

…



Recap of Adaboost

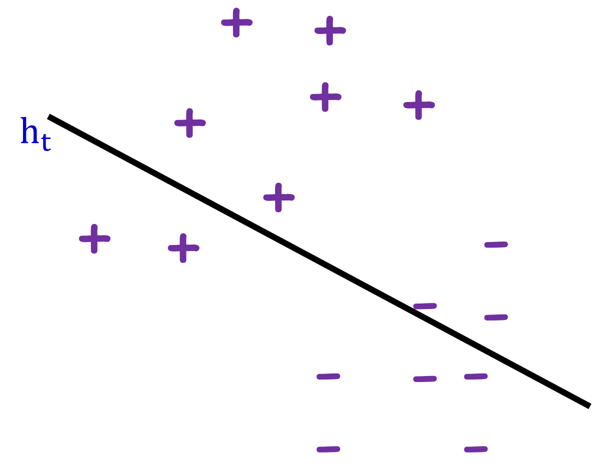
- Boosting: algorithmic technique for turning a weak learning algorithm into a strong (PAC) learning one.

Recap of Adaboost

- Boosting: turns a weak algo into a strong (PAC) learner.

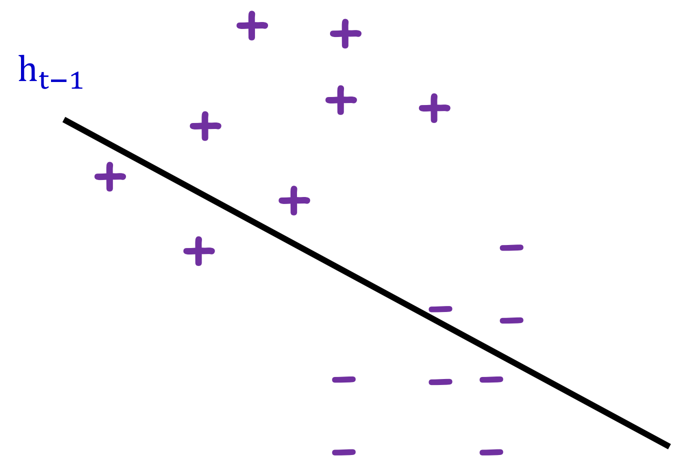
<u>Input</u>: $S=\{(x_1, y_1), \ldots,(x_m, y_m)\}$; weak learner A

- Weak learning algorithm A.
- For $t=1,2, \ldots ,T$
  - Construct D_t on $\{x_1, \ldots, x_m\}$
  - Run A on D_t producing h_t
- Output H_final=sgn($\sum \alpha_t \; h_t$)



Recap of Adaboost

- Weak learning algorithm A.
- For t=1,2, … ,T
  - **Construct D_t on $\{x_1, \ldots, x_m\}$**
  - Run A on D_t producing h_t



- D_1 uniform on $\{x_1, \ldots, x_m\}$
- D_{t+1} increases weight on x_i if h_t incorrect on x_i ; decreases it on x_i if h_t correct.

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \, e^{\{-\alpha_t\}} \text{ if } y_i = h_t(x_i)$$

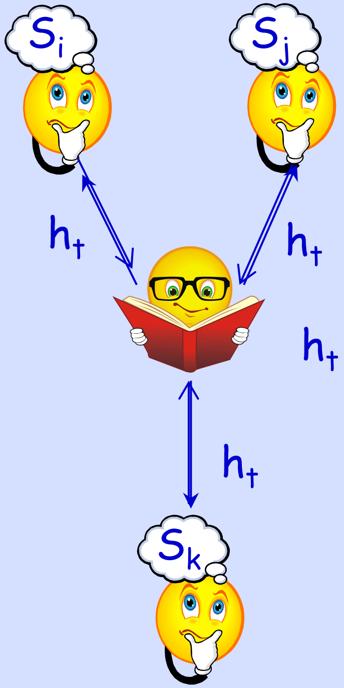
$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \, e^{\{\alpha_t\}} \text{ if } y_i \neq h_t(x_i)$$

<u>Key points</u>:

- $D_{t+1}(x_i)$ depends on $h_1(x_i), \ldots, h_t(x_i)$ and normalization factor that can be communicated efficiently.
- To achieve weak learning it suffices to use O(d) examples.

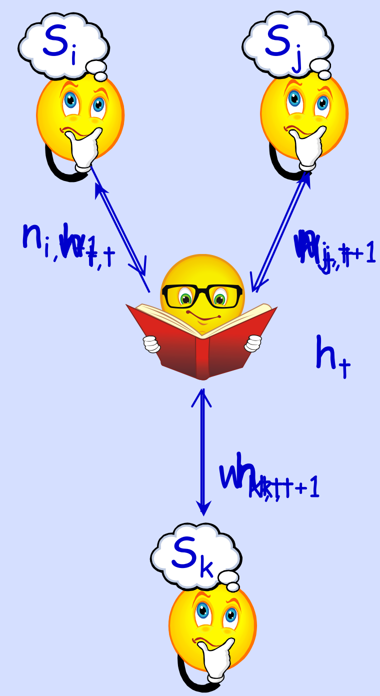
Distributed Adaboost

- Each player i has a sample S_i from D_i.
- For $t=1,2, \ldots ,T$
  - Each player sends player 1, enough data to produce weak hyp h_t.
    [For t=1, O(d/k) examples each.]
  - Player 1 broadcasts h_t to other players.



Distributed Adaboost

- Each player i has a sample S_i from D_i.
- For $t=1,2, \ldots ,T$
  - Each player sends player 1, enough data to produce weak hyp h_t.
    [For t=1, O(d/k) examples each.]
  - Player 1 broadcasts h_t to other players.
  - Each player i reweights its own distribution on S_i using h_t and sends the sum of its weights $w_{i,t}$ to player 1.
  - Player 1 determines the #of samples to request from each i [samples O(d) times from the multinomial given by $w_{i,t}/W_t$].



Distributed Adaboost

Can learn any class C with $O(\log(1/\epsilon))$ rounds using $O(d)$ examples + $O(k \log d)$ bits per round.

[efficient if can efficiently weak-learn from O(d) examples]

Proof:

- As in Adaboost, $O(\log 1/\epsilon)$ rounds to achieve error ϵ.
- Per round: $O(d)$ examples, $O(k \log d)$ extra bits for weights, 1 hypothesis.

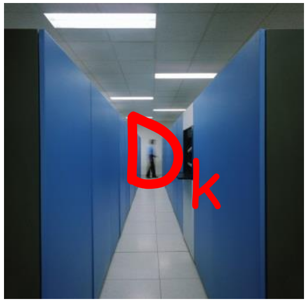
Dependence on $1/\epsilon$, Agnostic learning

Distributed implementation of Robust halving [Balcan-Hanneke'12].

- error $O(OPT)+\epsilon$ using only $O(k \log|C| \log(1/\epsilon))$ examples.

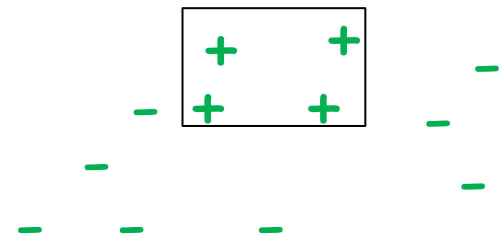
Not computationally efficient in general, but says $O(\log(1/\epsilon))$ possible in principle.

…



Better results for special cases

Intersection-closed when fns can be described compactly .



C is intersection-closed, then C can be learned in one round and k hypotheses of total communication.

Algorithm:

- Each i draws S_i of size $O(d/\epsilon \log(1/\epsilon))$, finds smallest h_i in C consistent with S_i and sends h_i to player 1.
- Player 1 computes smallest h s.t. $h_i \subseteq h$ for all i.

<u>Key point</u>:

h_i, h never make mistakes on negatives, so $\mathrm{err}_{D_i}(h) \le \mathrm{err}_{D_i}(h_i) \le \epsilon$.

Better results for special cases

E.g., conjunctions over $\{0,1\}^d$ [$f(x) = x_2 x_5 x_9 x_{15}$]

- Only O(k) examples sent, O(kd) bits.
  - Each entity intersects its positives.
  - Sends to player 1.
  - Player 1 intersects & broadcasts.

1101111011010111
1111110111001110
1100110011001111
1100110011000110

[Generic methods O(d) examples, or $O(d^2)$ bits total.]

Interesting class: parity functions

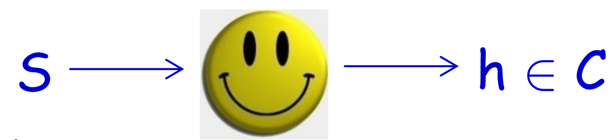
- $k = 2, X = \{0,1\}^d$, C = parity fns, $f(x) = x_{i_1} \text{XOR } x_{i_2} \ldots \text{XOR } x_{i_l}$
- Generic methods: O(d) examples, O(d^2) bits.
- Classic CC lower bound: $\Omega(d^2)$ bits LB for proper learning.

Improperly learn C with O(d) bits of communication!

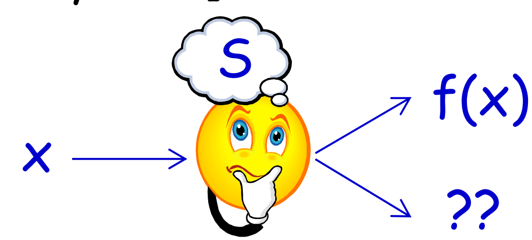
<u>Key points</u>:

- Can properly PAC-learn C.

  [Given dataset S of size O(d/ϵ), just solve the linear system]



- Can non-properly learn C in reliable-useful manner [RS'88]



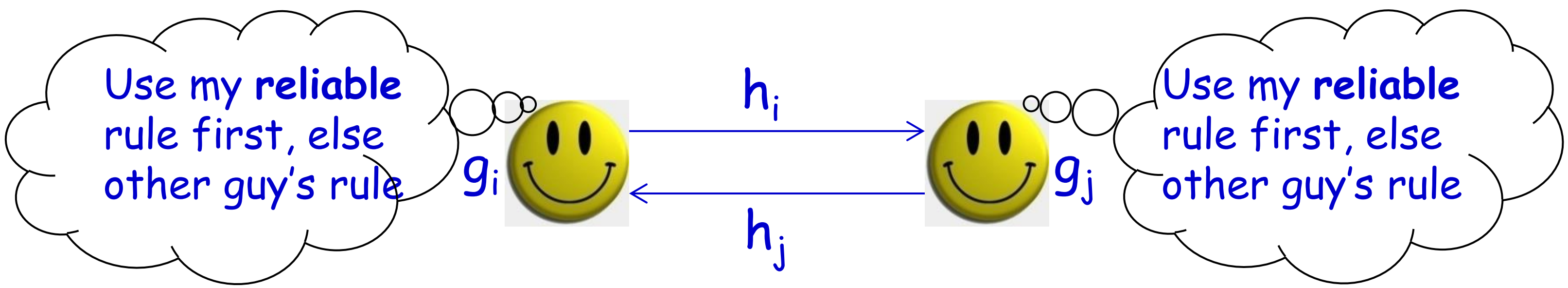
[if x in subspace spanned by S, predict accordingly, else say "?"]

Interesting class: parity functions

Improperly learn C with $O(d)$ bits of communication!

Algorithm:

- Player i properly PAC-learns over D_i to get parity h_i. Also improperly R-U learns to get rule g_i. Sends h_i to player j.
- Player i uses rule R_i: "if g_i predicts, use it; else use h_j"



<u>Key point</u>: low error under D_j because h_j has low error under D_j and since g_i never makes a mistake putting it in front does not hurt.

Distributed PAC learning: Summary

- Communication as a fundamental resource.
- General bounds on communication, communication-efficient distributed boosting.
- Improved bounds for special classes (intersection-closed, parity fns, and linear separators over nice distributions).

Open questions

- Efficient algorithms in noisy settings.
- Other learning tasks.
  - k-Means and k-Median Clustering on General Topologies [Balcan-Ehrlich-Liang, NIPS 2013]
- More refined trade-offs between communication complexity, computational complexity, and sample complexity.